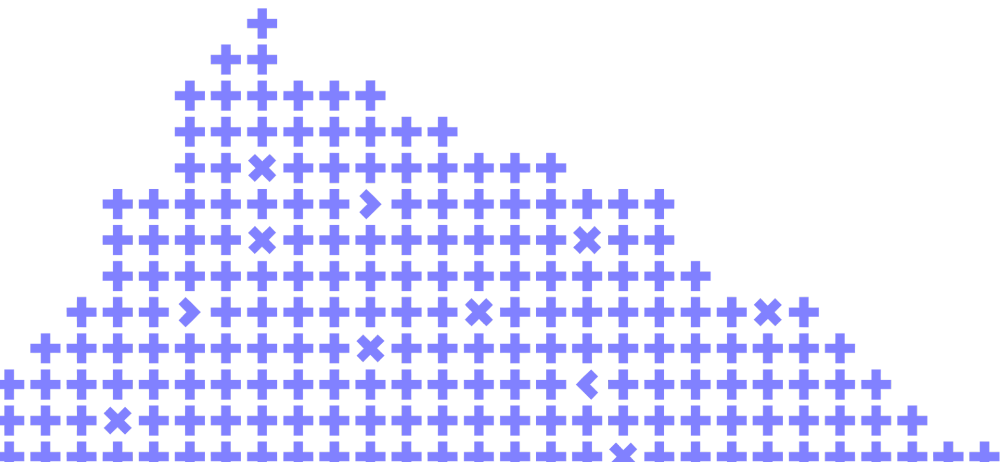


Push notifications in RuStore: how we built an alternative transport to replace Google Firebase

Kirill Alekseev



Co-organizer

Yandex

About me

2



About me

2

- Got the bachelor's degree and the master's degree of the MSU Faculty of Computational Mathematics and Cybernetics



About me

- Got the bachelor's degree and the master's degree of the MSU Faculty of Computational Mathematics and Cybernetics
- At VK I manage several backend development teams in Mail.ru Mail Service



2



About me

2

- Got the bachelor's degree and the master's degree of the MSU Faculty of Computational Mathematics and Cybernetics
- At VK I manage several backend development teams in Mail.ru Mail Service
- Playing guitar and planning sprints at "Dela Povazhnee"



**RuStore is being
developed with the
contribution of the largest
Russian IT companies.**

Thank you
for your
contacts!

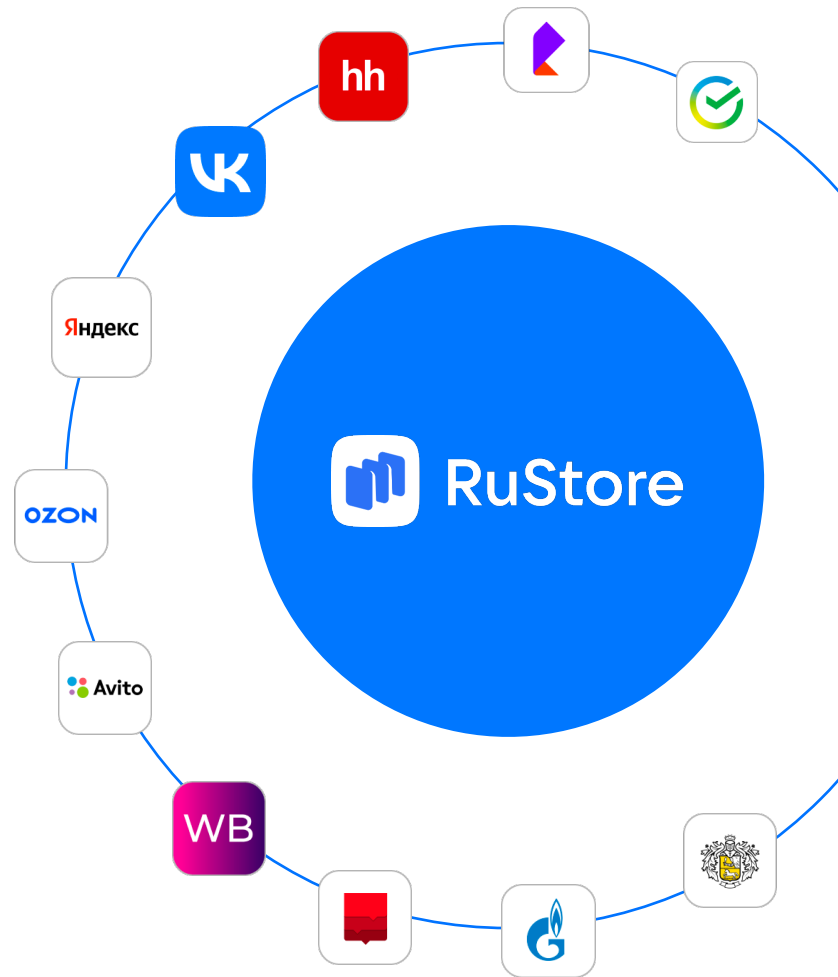


Table of Contents



Table of Contents

- Introduction



Table of Contents



Introduction



Overview of the
service's public part



Table of Contents



Introduction



Overview of the
service's public part



Internals of our SDK

Table of Contents



Introduction



Overview of the
service's public part



Internals of our SDK



Push notification service
backend architecture
details

Table of Contents



Introduction



Overview of the
service's public part



Internals of our SDK



Push notification service
backend architecture
details



Why we
won't be down (at least
for too long)

Table of Contents



Introduction



Overview of the
service's public part



Why we
won't be down (at least
for too long)



Internals of our SDK



How we integrated
RuStore push notifications
into Mail.ru Mail Service
(and what came out of it)



Push notification service
backend architecture
details

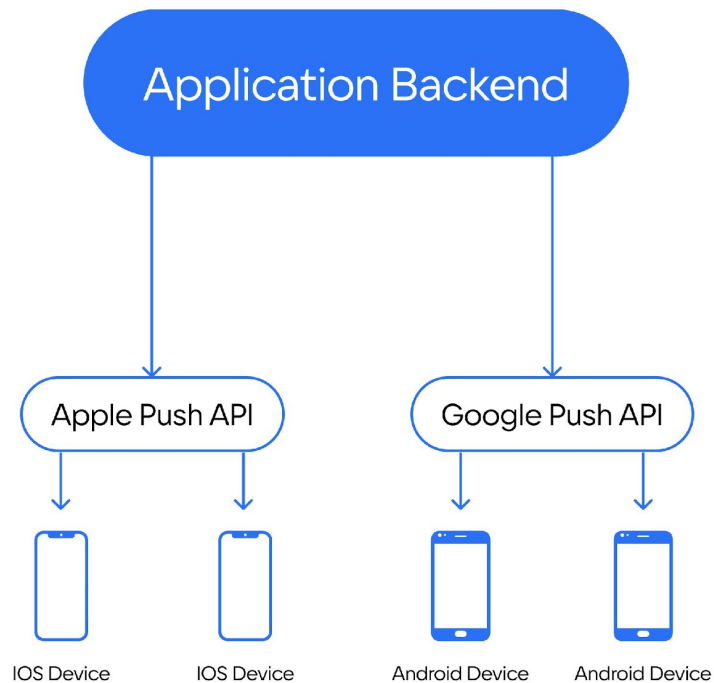


RuStore



Introduction

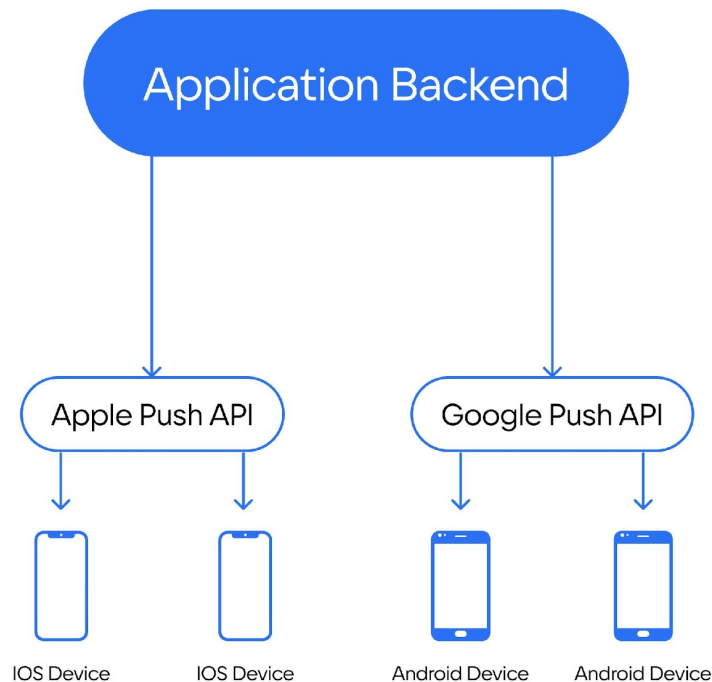
Why build our own transport of push notifications



6

Why build our own transport of push notifications

Push notifications are sent via Google/Apple/Huawei API

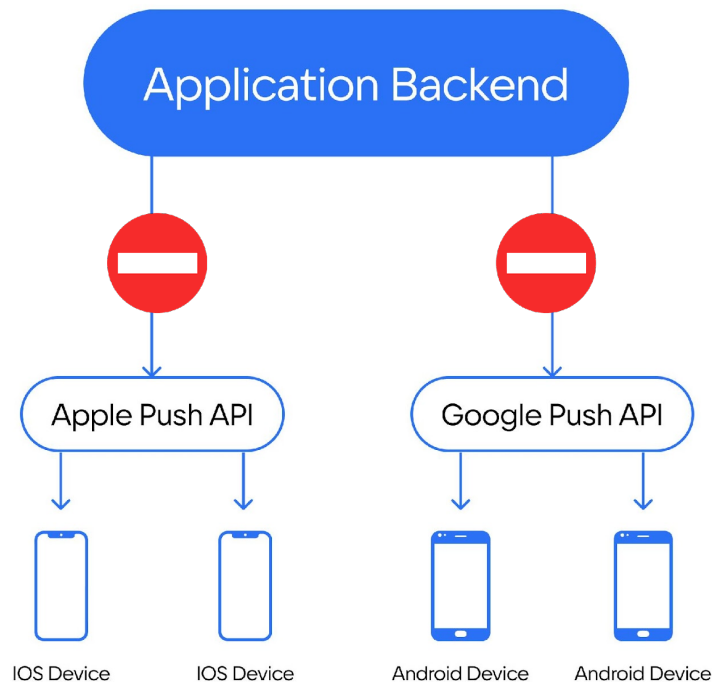


6

Why build our own transport of push notifications

Push notifications are sent via Google/Apple/Huawei API

If Google decides that we no longer need push notifications, then we no longer have it



6

Requirements for the service

7



Requirements for the service



Possibility to
subscribe
and unsubscribe
from push notifications



Requirements for the service



Possibility to
subscribe
and unsubscribe
from push notifications



Possibility to send push
notifications for a given token
with instant delivery
to a device



Requirements for the service

7

- Possibility to subscribe and unsubscribe from push notifications
- Possibility to send push notifications for a given token with instant delivery to a device
- Temporary storage of push notifications while the device is offline (with TTL)

Requirements for the service

7

- Possibility to subscribe and unsubscribe from push notifications
- Possibility to send push notifications for a given token with instant delivery to a device
- Temporary storage of push notifications while the device is offline (with TTL)
- Service must be reliable

Requirements for the service

7

- Possibility to subscribe and unsubscribe from push notifications
- Possibility to send push notifications for a given token with instant delivery to a device
- Temporary storage of push notifications while the device is offline (with TTL)
- Service must be reliable
- Service must be secure

Requirements for the service

7

- Possibility to subscribe and unsubscribe from push notifications
- Possibility to send push notifications for a given token with instant delivery to a device
- Temporary storage of push notifications while the device is offline (with TTL)
- Service must be reliable
- Service must be secure
- Easy integration into the current scheme of working with push notifications

What any push
notification looks like

8



What any push notification looks like



Push token identifies the application on a specific device

```
{  
  "token": "PFGUGOBfCPZe3U0FU3Uf1V8qPGUddEBD",  
  "title": "Kirill Alekseev\nThis is a push title",  
  "body": "This is a push body",  
  "some_other_useful_data": "42"  
}
```

What any push notification looks like

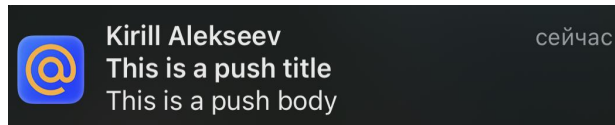


Push token identifies the application on a specific device



In a push notification, you can send an arbitrary payload and process it client-side

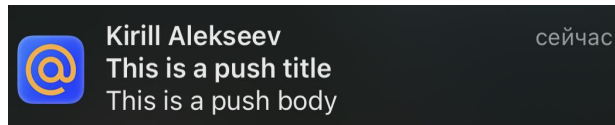
```
{  
  "token": "PFGUGOBfCPZe3U0FU3Uf1V8qPGUddeBD",  
  "title": "Kirill Alekseev\nThis is a push title",  
  "body": "This is a push body",  
  "some_other_useful_data": "42"  
}
```



What any push notification looks like

- Push token identifies the application on a specific device
- In a push notification, you can send an arbitrary payload and process it client-side
- Built-in push notification transports use internal OS **mechanisms that are inaccessible to us**

```
{  
  "token": "PFGUGOBfCPZe3U0FU3Uf1V8qPGUddeBD",  
  "title": "Kirill Alekseev\nThis is a push title",  
  "body": "This is a push body",  
  "some_other_useful_data": "42"  
}
```





RuStore




Overview of the service's public part

A web-interface to manage push notifications of application in RuStore

10

← All apps

 **Маруся**

Versions

Users and permissions

Reviews

Application statistics

▼ Monetization Beta

Subscriptions


In-app purchases

Manage payments

▼ Push-notifications Beta

Projects


Projects > Prod PROD


Project ID
avsf3jbvlwq38y03 


Signature imprint SHA-256
91:AC:3E:2F:CB:C2:42:87:6A:85:02:86:EB:44:84:10:34:02:ED:35:CE:C6:38:47:E
F:50:07:2B:E0:D9:8D:8B


Service tokens

Generated: 2/5 + New token

★ r409-jr0ijfwej 



☆ r_409i0r3226 




The logo for High Load Armenia, featuring the text "High Load" in a bold, sans-serif font with a red double-plus sign above it, and "Armenia" in a smaller font below. The entire logo is contained within a blue hexagonal shape.

A web-interface to manage push notifications of application in RuStore

10

Any application is identified with the project ID

[← All apps](#)

 **Маруся**

[☐ Versions](#)

[⚙ Users and permissions](#)

[🗨 Reviews](#)

[📈 Application statistics](#)

▼ Monetization Beta

[📄 Subscriptions](#)

[🛒 In-app purchases](#)

[💳 Manage payments](#)

▼ Push-notifications Beta

[🔗 Projects](#)

Projects > Prod PROD

Project ID

avsf3jbvlwq38y03 [📄](#)

Signature imprint SHA-256

91:AC:3E:2F:CB:C2:42:87:6A:85:02:86:EB:44:84:10:34:02:ED:35:CE:C6:38:47:E
F:50:07:2B:E0:D9:8D:8B

Service tokens

Generated: 2/5

[+ New token](#)

★ r409-jr0ijfwej [📄](#)

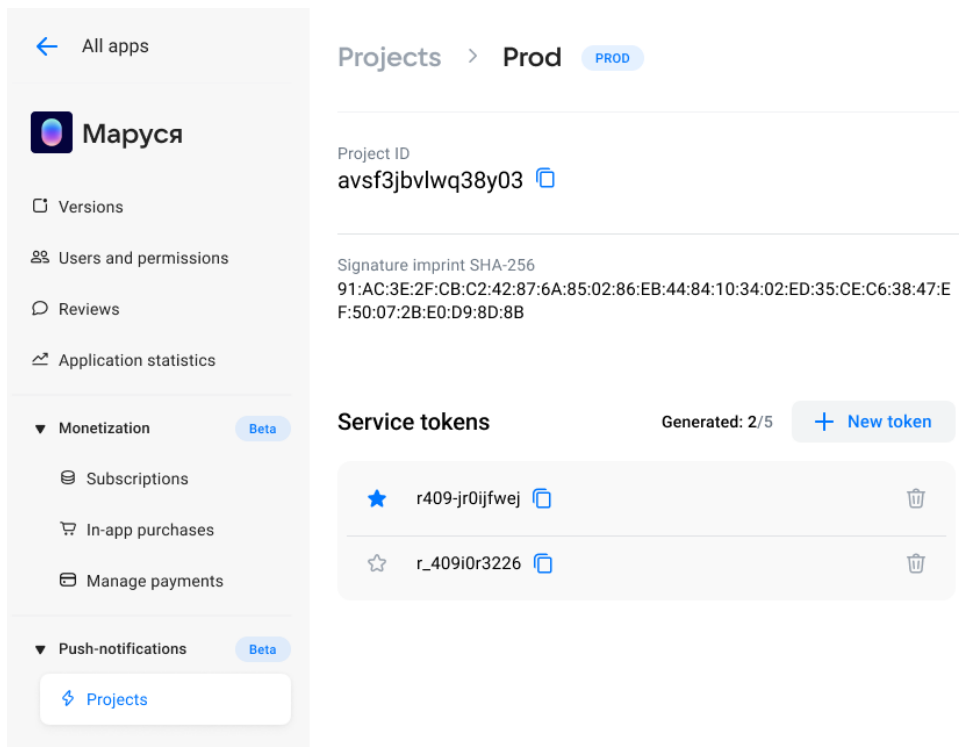


☆ r_409i0r3226 [📄](#)



A web-interface to manage push notifications of application in RuStore

10



Any application is identified with the project ID


Service tokens authorize sending a push notification **and are sensitive**



A web-interface to manage push notifications of application in RuStore

11

← All apps

 **Маруся**

☐ Versions

⚙ Users and permissions

💬 Reviews

📈 Application statistics

▼ Monetization Beta

📄 Subscriptions




🛒 In-app purchases

💳 Manage payments

▼ Push-notifications Beta

⚡ Projects

Projects


Name ↑	Project ID ⓘ
Dev	24526fbsr0jfw032 
Prod PROD	avsf3jbvlwq38y03 
Test 1	r409jr0ijfweji0r32 



A web-interface to manage push notifications of application in RuStore

11

[← All apps](#)

 **Маруся**

[☐ Versions](#)

[⚙ Users and permissions](#)

[💬 Reviews](#)

[📈 Application statistics](#)

▼ Monetization Beta

[📄 Subscriptions](#)




[🛒 In-app purchases](#)

[💰 Manage payments](#)

▼ Push-notifications Beta

[⚡ Projects](#)

Projects

Name ↑	Project ID ⓘ
Dev	24526fbsr0ijfwi032 
Prod PROD	avsf3jbvlwq38y03 
Test 1	r409jr0ijfweji0r32 

You can isolate the Dev and the Prod service tokens by linking several push projects to one application



SDK for push service in RuStore

12



SDK for push service in RuStore

12



We try to be a drop-in
replacement for FCM
SDK



SDK for push service in RuStore



We try to be a drop-in replacement for FCM SDK



The getToken method registers and gives the application a new push token

SDK for push service in RuStore



We try to be a drop-in replacement for FCM SDK



The getToken method registers and gives the application a new push token



The deleteToken method invalidates an existing token

SDK for push service in RuStore



We try to be a drop-in replacement for FCM SDK



The getToken method registers and gives the application a new push token



The deleteToken method invalidates an existing token



You can define callbacks onMessageReceived, onNewToken, onDeletedMessages

RuStore Push Service API Methods

13

```
curl --XPOST 'https://vkpns.rustore.ru/v1/projects/admlkdas/messages:send' \  
  --header 'Authorization: Bearer dasmlkads' --header 'Content-Type: application/json' \  
  --data-raw '{"message":{  
    "token":"PFGUG0BfCPZe3U0FU3UflV8qPGUddEBD",  
    "android":{ "notification":{ "title":"Hello, world!"} }  
  }}'
```



RuStore Push Service API Methods



We try to be a drop-in replacement for the FCM API

```
curl --XPOST 'https://vkpns.rustore.ru/v1/projects/admlkdas/messages:send' \  
  --header 'Authorization: Bearer dasmlkads' --header 'Content-Type: application/json' \  
  --data-raw '{"message":{  
    "token":"PFGUG0BfCPZe3U0FU3UflV8qPGUddEBD",  
    "android":{ "notification":{ "title":"Hello, world!"} }  
  }}'
```

RuStore Push Service API Methods



We try to be a drop-in replacement for the FCM API



We support both notification and data pushes, but so far a limited set of fields

```
curl --XPOST 'https://vkpns.rustore.ru/v1/projects/admlkdas/messages:send' \  
  --header 'Authorization: Bearer dasmlkads' --header 'Content-Type: application/json' \  
  --data-raw '{"message":{  
    "token":"PFGUG0BfCPZe3U0FU3UflV8qPGUddEBD",  
    "android":{ "notification":{ "title":"Hello, world!"} }  
  }}'
```



RuStore



Internals
of our SDK

Main SDK components

15



Main SDK components



There are two SDKs:
"client" and "host"

Main SDK components



There are two SDKs:
"client" and "host"



"Host" is built into
RuStore and collects
**all push notifications for
a given device**

Main SDK components



There are two SDKs:
"client" and "host"



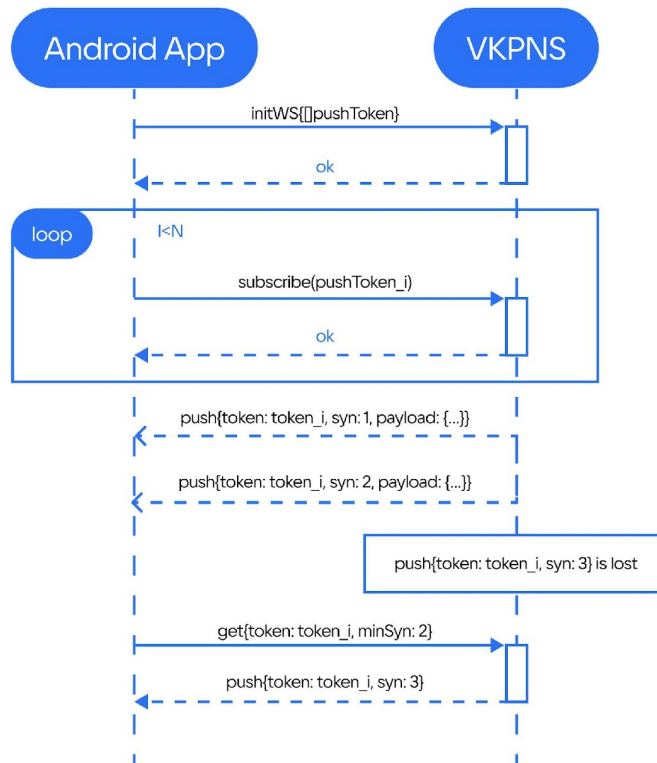
"Host" is built into
RuStore and collects
**all push notifications for
a given device**



"Client" must be integrated
by the application developer
(similar to the Firebase
SDK)

How SDK receives pushes

Most notifications come instantly via WebSocket, but HTTP polling is also performed periodically to deliver the lost ones.



Fetching notifications in the background

Background instant delivery
of push notifications requires
a permission from the user

Let app always run in
background?

Allowing RuStore to always run in the
background may reduce battery life.

You can change this later from Settings >
Apps.

Deny

Allow



RuStore



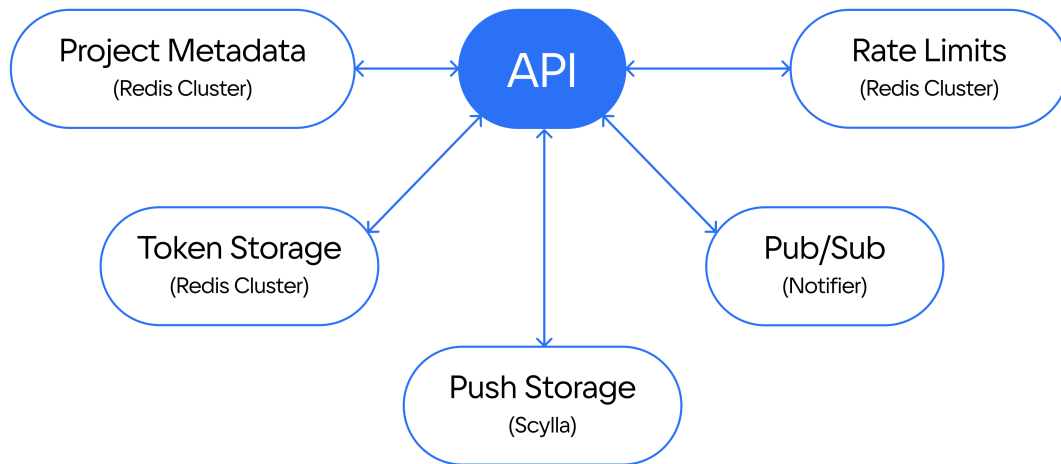
Push service
backend
architecture
details

Main components of the backend

Golang API in k8s

Redis Cluster

Scylla



Push token storage overview

20



Push token storage overview

20

Stored in Redis
Cluster

```
hgetall token:{8B83XYXrmCYB9xF0mozIk-uZN}  
1) "project_id"  
2) "9TLrisVIsNfEc0DjE4clbJYsyyG9ACs4"  
3) "syn"  
4) "145"  
7) "syn_stored"  
8) "145"
```



Push token storage overview

20

Stored in Redis
Cluster

SYN — monotonically
increasing push counter

```
hgetall token:{8B83XYXrmCYB9xF0mozIk-uZN}
```

```
1) "project_id"
```

```
2) "9TLrisVIsNfEc0DjE4clbJYsyyG9ACs4"
```

```
3) "syn"
```

```
4) "145"
```

```
7) "syn_stored"
```

```
8) "145"
```



Push token storage overview

20

Stored in Redis
Cluster

SYN — monotonically
increasing push counter

Requires less than 400
MB for 1 million tokens

```
hgetall token:{8B83XYXrmCYB9xF0mozIk-uZN}
```

```
1) "project_id"
```

```
2) "9TLrisVIsNfEc0DjE4clbJYsyyG9ACs4"
```

```
3) "syn"
```

```
4) "145"
```

```
7) "syn_stored"
```

```
8) "145"
```



Push storage overview

21



Push storage overview

21

Stored in Scylla
on disk

```
CREATE TABLE vkpns.push (  
    token_id text,  
    syn bigint,  
    payload text,  
    timestamp timestamp,  
    PRIMARY KEY (token_id, syn)  
) WITH CLUSTERING ORDER BY (syn ASC)  
    AND bloom_filter_fp_chance = 0.01  
    AND compaction = {'class': 'SizeTieredCompactionStrategy'}  
    AND compression = {'sstable_compression': 'LZ4Compressor'};
```

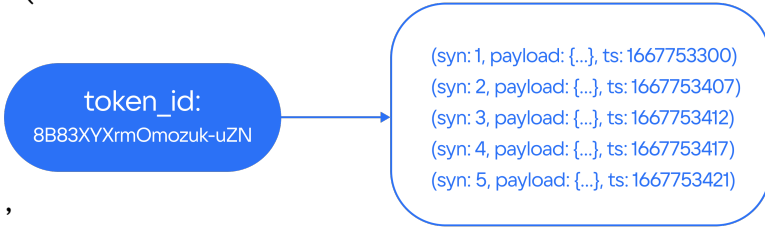


Push storage overview

21

Stored in Scylla
on disk

```
CREATE TABLE vkpns.push (  
  token_id text,  
  syn bigint,  
  payload text,  
  timestamp timestamp,  
  PRIMARY KEY (token_id, syn)  
) WITH CLUSTERING ORDER BY (syn ASC)  
  AND bloom_filter_fp_chance = 0.01  
  AND compaction = {'class': 'SizeTieredCompactionStrategy'}  
  AND compression = {'sstable_compression': 'LZ4Compressor'};
```



The diagram illustrates the relationship between a `token_id` and its corresponding records in the `vkpns.push` table. A blue rounded rectangle labeled `token_id:` contains the value `8B83XYXrmOmozuk-uZN`. An arrow points from this rectangle to a larger rounded rectangle containing a list of five records, each with a `syn` value, a `payload`, and a `ts` (timestamp).

- (syn: 1, payload: {...}, ts: 1667753300)
- (syn: 2, payload: {...}, ts: 1667753407)
- (syn: 3, payload: {...}, ts: 1667753412)
- (syn: 4, payload: {...}, ts: 1667753417)
- (syn: 5, payload: {...}, ts: 1667753421)

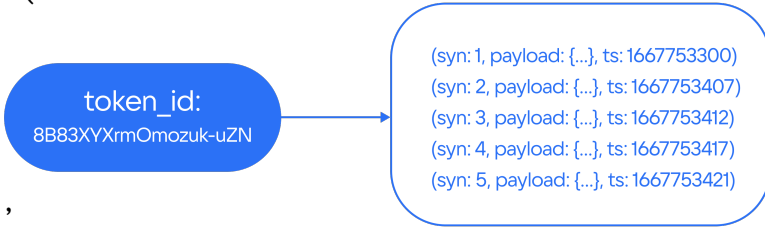
Push storage overview

21

Stored in Scylla
on disk

Allows efficient reading
by token+syn_range

```
CREATE TABLE vkpns.push (  
    token_id text,  
    syn bigint,  
    payload text,  
    timestamp timestamp,  
    PRIMARY KEY (token_id, syn)  
) WITH CLUSTERING ORDER BY (syn ASC)  
    AND bloom_filter_fp_chance = 0.01  
    AND compaction = {'class': 'SizeTieredCompactionStrategy'}  
    AND compression = {'sstable_compression': 'LZ4Compressor'};
```



The diagram illustrates the relationship between a token_id and its corresponding syn values. A blue rounded rectangle labeled "token_id: 8B83XYXrmOmozuk-uZN" has an arrow pointing to a larger rounded rectangle containing a list of five entries, each with a syn value, a payload, and a timestamp.

syn	payload	ts
1	{...}	1667753300
2	{...}	1667753407
3	{...}	1667753412
4	{...}	1667753417
5	{...}	1667753421

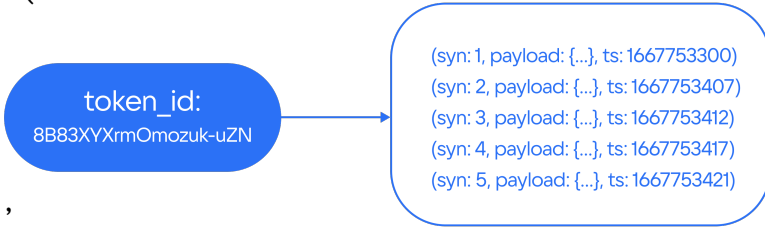
Push storage overview

Stored in Scylla
on disk

Allows efficient reading
by token+syn_range

LWW (Last Write Wins)
saves us from duplicate
push notifications

```
CREATE TABLE vkpns.push (  
  token_id text,  
  syn bigint,  
  payload text,  
  timestamp timestamp,  
  PRIMARY KEY (token_id, syn)  
) WITH CLUSTERING ORDER BY (syn ASC)  
  AND bloom_filter_fp_chance = 0.01  
  AND compaction = {'class': 'SizeTieredCompactionStrategy'}  
  AND compression = {'sstable_compression': 'LZ4Compressor'};
```



The diagram illustrates the relationship between a token_id and its corresponding syn and timestamp values. A blue rounded rectangle labeled "token_id: 8B83XYXrmOmozuk-uZN" has an arrow pointing to a larger rounded rectangle containing a list of five entries, each with a syn value, a payload (represented by {...}), and a timestamp (ts).

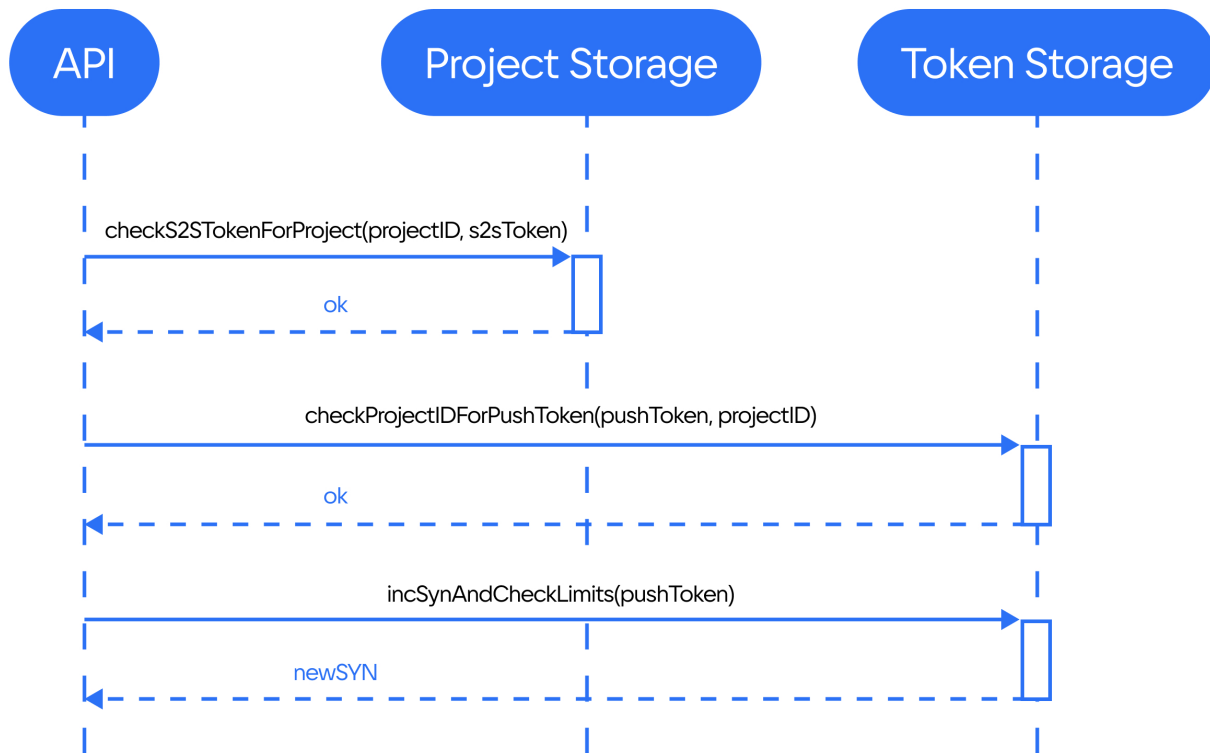
syn	payload	ts
1	{...}	1667753300
2	{...}	1667753407
3	{...}	1667753412
4	{...}	1667753417
5	{...}	1667753421

API logic for sending a push notification

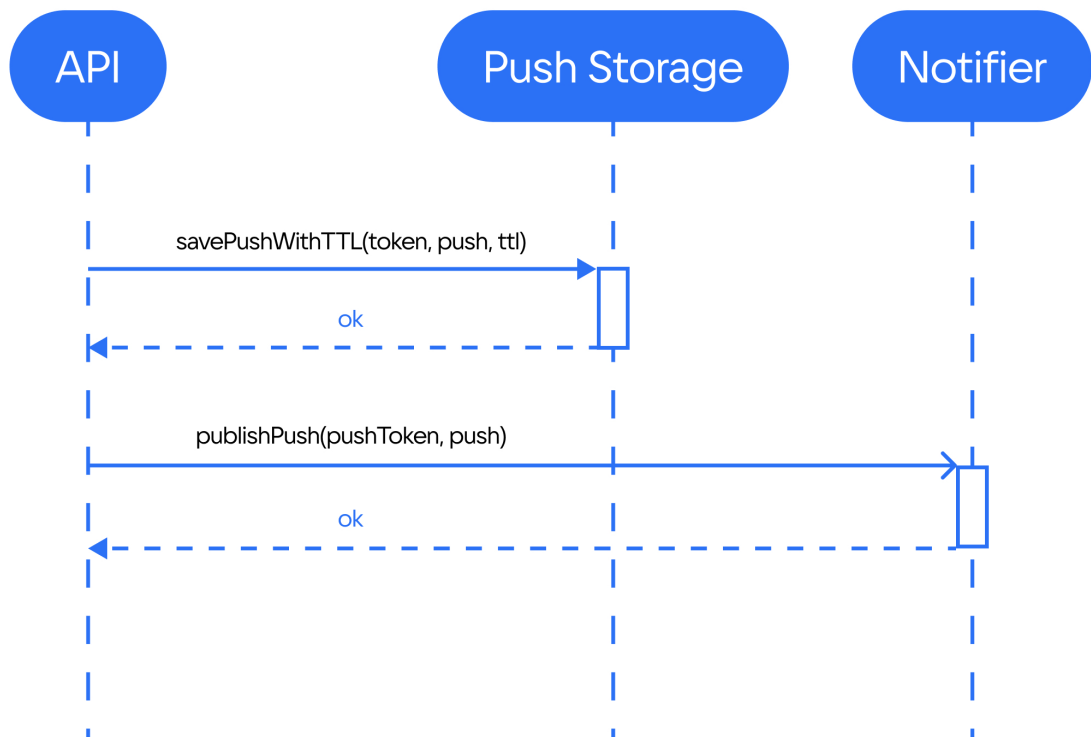
```
curl --XPOST 'https://vkpns.rustore.ru/v1/projects/admlkdas/messages:send' \  
  --header 'Authorization: Bearer dasmlkads' --header 'Content-Type: application/json' \  
  --data-raw '{"message":{  
    "token":"PFGUGOBfCPZe3U0FU3UflV8qPGUddEBD",  
    "android":{"notification":{"title":"Hello, world!"}} }  
  }}'
```

API logic for sending a push notification

23



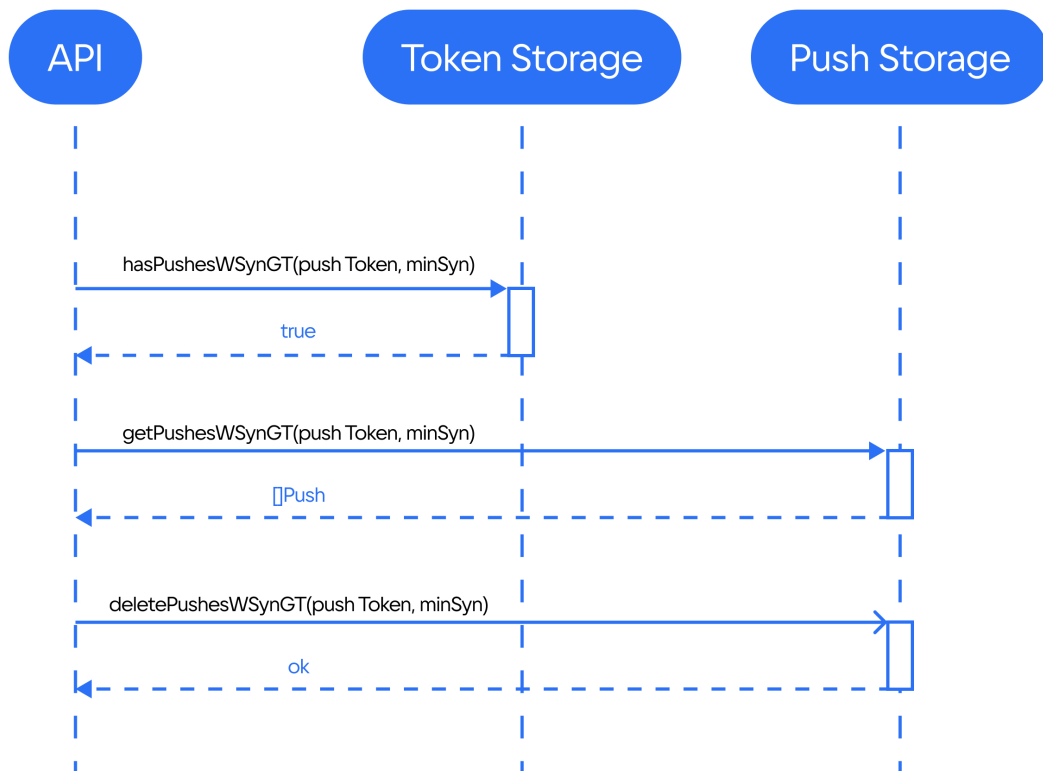
API logic for sending a push notification



API logic for receiving a push notification

```
curl 'https://vkpns.rustore.ru/v1/projects/messages:get' -H'Content-Type: application/json' \
  --data-raw '{ "requests": [
    {
      "project_id": "admlkdas", "min_syn": 12, "limit": 10,
      "token": "PFGUG0BfCPZe3U0FU3Uf1V8qPGUddeBD"
    }
  ]}'
```


API logic for receiving a push notification



How not to overload the storage
with stale push tokens

27



How not to overload the storage with stale push tokens

27



Reasons why stale
tokens appear:

How not to overload the storage with stale push tokens



Reasons why stale tokens appear:

app has been removed

How not to overload the storage with stale push tokens



Reasons why stale tokens appear:

app has been removed

phone got lost

How not to overload the storage with stale push tokens



Reasons why stale tokens appear:

app has been removed

phone got lost

???

How not to overload the storage with stale push tokens



Reasons why stale tokens appear:

app has been removed

phone got lost

???



All push tokens have a TTL in the token storage

How not to overload the storage with stale push tokens



Reasons why stale tokens appear:

app has been removed

phone got lost

???



All push tokens have a TTL in the token storage



Automatically update the lifetime of the token, if the client comes for pushes



RuStore



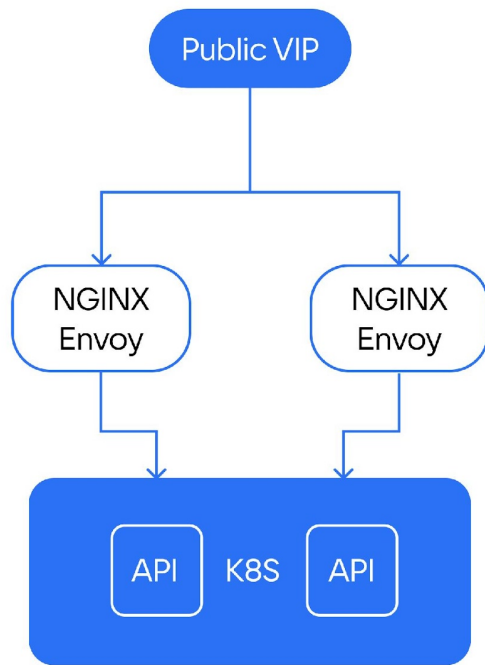
Why we won't
be down (at least
for too long)

Fault tolerance at the API level

API lives in Kubernetes

Kubernetes lives in 5 DCs

Pod healthchecks configured in Envoy



29

Redis Cluster for Storage of push tokens

30



Redis Cluster for Storage of push tokens

30



Scales horizontally
thanks to:



Redis Cluster for Storage of push tokens

30



Scales horizontally
thanks to:

- automatic resharding



Redis Cluster for Storage of push tokens



Scales horizontally
thanks to:

- automatic resharding
- routing of read queries to replicas

Redis Cluster for Storage of push tokens



Scales horizontally
thanks to:

- automatic resharding
- routing of read queries to replicas



Fault-tolerant in case of a
node/datacenter failure thanks to:

Redis Cluster for Storage of push tokens



Scales horizontally
thanks to:

- automatic resharding
- routing of read queries to replicas



Fault-tolerant in case of a
node/datacenter failure thanks to:

- automatic failover
(with short downtime)

Redis Cluster for Storage of push tokens



Scales horizontally thanks to:

- automatic resharding
- routing of read queries to replicas



Fault-tolerant in case of a node/datacenter failure thanks to:

- automatic failover (with short downtime)
- retries of read queries to another replica

Scylla for Storage of pushes

31



Scylla for Storage of pushes

31



Scales horizontally with
automatic resharding



Scylla for Storage of pushes

31



Scales horizontally with automatic resharding



Fault-tolerant in case of a node/datacenter failure thanks to the "no master" architecture



Scylla for Storage of pushes

31



Scales horizontally with automatic resharding



Fault-tolerant in case of a node/datacenter failure thanks to the "no master" architecture



With no downtime!



Scylla for Storage of pushes

31



Scales horizontally with automatic resharding



Fault-tolerant in case of a node/datacenter failure thanks to the "no master" architecture



With no downtime!



~~Million~~

10 million WebSockets and Go

32



~~Million~~

10 million WebSockets and Go

32



The service has been
successfully operated
in Mail.ru Mail Service
for five years



~~Million~~

10 million WebSockets and Go

32



The service has been successfully operated in Mail.ru Mail Service for five years



Resistant to one DC failure



~~Million~~

10 million WebSockets and Go

32



The service has been successfully operated in Mail.ru Mail Service for five years

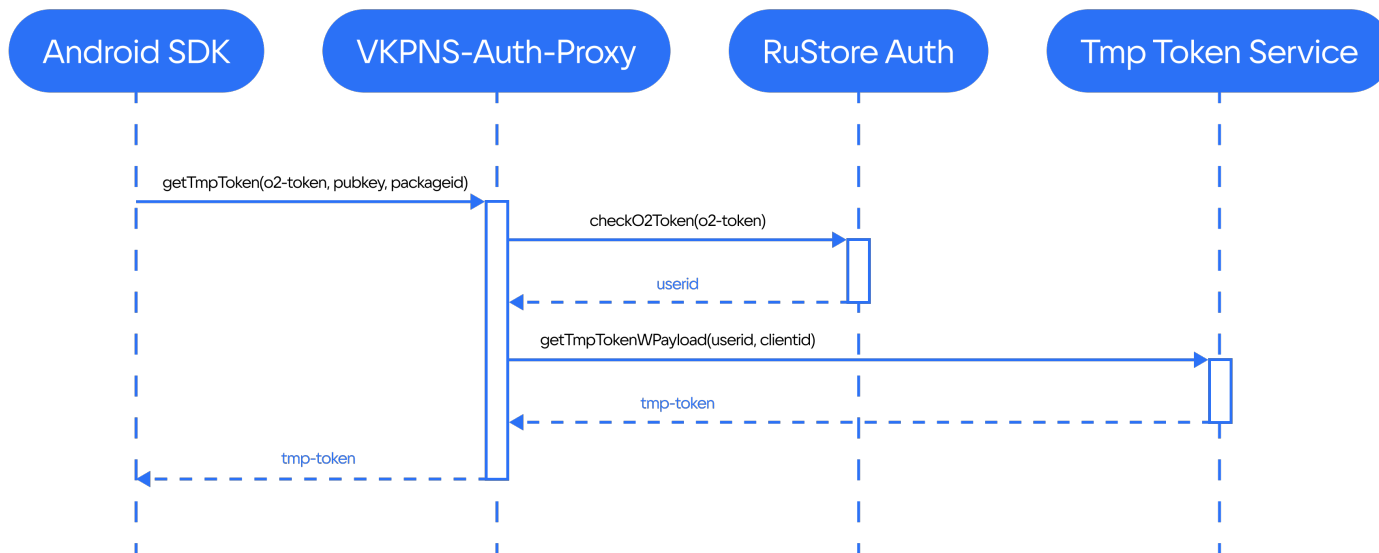


Resistant to one DC failure



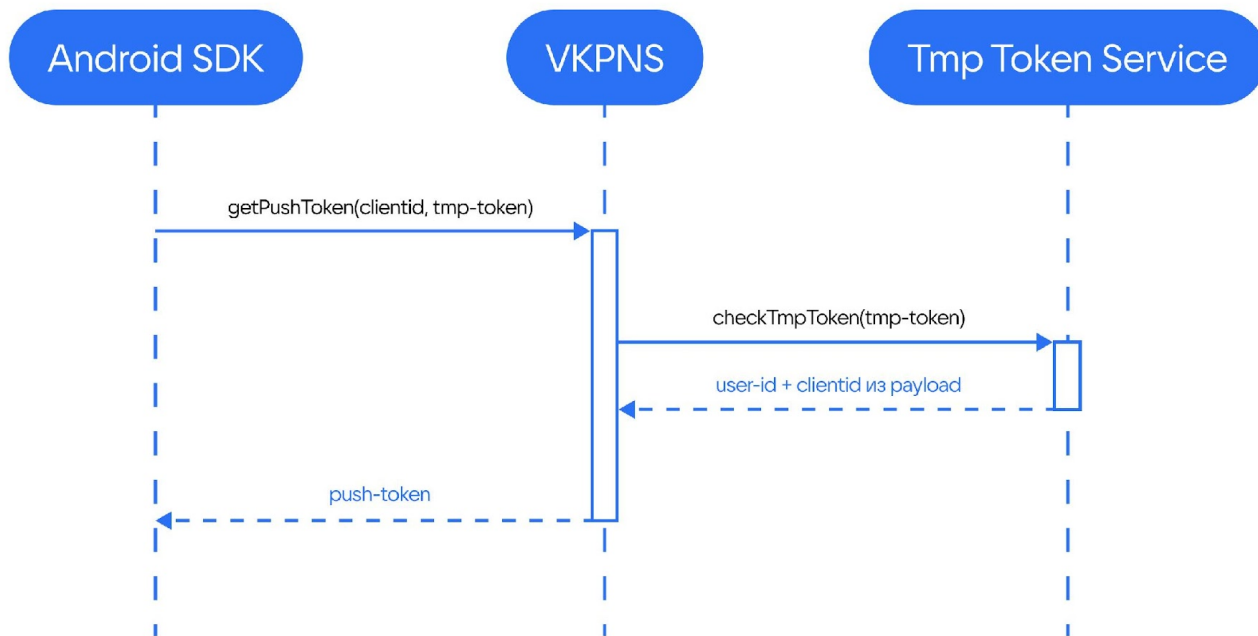
DoS protection on generation of push tokens

33



DoS protection on generation of push tokens

34





RuStore



How we integrated
RuStore push
notifications into
Mail.ru Mail Service

Integration process to Mail.ru Mail Service

36



Integration process to Mail.ru Mail Service

36



Started sending
push notifications
through RuStore
in parallel with FCM



Integration process to Mail.ru Mail Service

36

- Started sending push notifications through RuStore in parallel with FCM

- Measured the number of delivered push notifications per hour



Integration process to Mail.ru Mail Service

36

- Started sending push notifications through RuStore in parallel with FCM
- Measured the number of delivered push notifications per hour
- Fixed bugs in the push service and analytics



Integration process to Mail.ru Mail Service

36

- Started sending push notifications through RuStore in parallel with FCM
- Measured the number of delivered push notifications per hour
- Fixed bugs in the push service and analytics
- Will start showing the push that came before



Integration process to Mail.ru Mail Service

36

- Started sending push notifications through RuStore in parallel with FCM
- Measured the number of delivered push notifications per hour
- Fixed bugs in the push service and analytics
- Will start showing the push that came before
- Will measure delivery latency



Integration process to Mail.ru Mail Service

- Started sending push notifications through RuStore in parallel with FCM
- Measured the number of delivered push notifications per hour
- Fixed bugs in the push service and analytics
- Will start showing the push that came before
- Will measure delivery latency
- Will reduce delivery latency



Complexity of integration

37

Complexity of integration

37



To enable push notifications, you need to make just a couple of clicks in the RuStore developer console



Complexity of integration

37



To enable push notifications, you need to make just a couple of clicks in the RuStore developer console



The interface of our SDK repeats the interface of the FCM SDK



Complexity of integration

37

- To enable push notifications, you need to make just a couple of clicks in the RuStore developer console
- The interface of our SDK repeats the interface of the FCM SDK
- The push notification sending API is a drop-in replacement for the FCM API



Benefits of integration

38



Benefits of integration

38

- Protected in case
Google blocks
our pushes



Benefits of integration

38

- Protected in case Google blocks our pushes

- We can increase the delivery rate by sending via two channels in parallel



Benefits of integration

38

- Protected in case Google blocks our pushes

- We can increase the delivery rate by sending via two channels in parallel

- Carried out load testing of RuStore push notifications in real conditions



Midterm results (VKPNS vs FCM)

39

Midterm results (VKPNS vs FCM)

39



Number of successfully
sent push notifications per
day: 99.999% vs 99.993%



Midterm results (VKPNS vs FCM)

39

- Number of successfully sent push notifications per day: 99.999% vs 99.993%

- API send average response time: 13ms vs 45ms



Midterm results (VKPNS vs FCM)

39

- Number of successfully sent push notifications per day: 99.999% vs 99.993%

- API send average response time: 13ms vs 45ms

- Push notification deliverability per day: ~101% vs 100%



Future plans

40



Future plans

40



Get closer in functionality to FCM (broadcasting by topics, more push settings, show deliverability metrics)



Future plans

40

- Get closer in functionality to FCM (broadcasting by topics, more push settings, show deliverability metrics)

- Start receiving push notifications through other VK applications on the user's device



Future plans

40

- Get closer in functionality to FCM (broadcasting by topics, more push settings, show deliverability metrics)
- Start receiving push notifications through other VK applications on the user's device
- Maintain health metrics no worse than that of Google FCM



Leave your feedback!

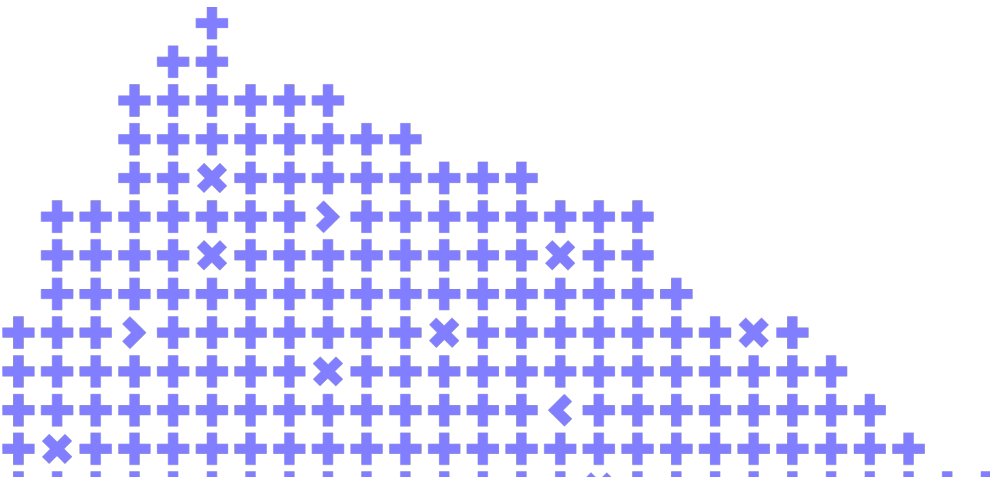
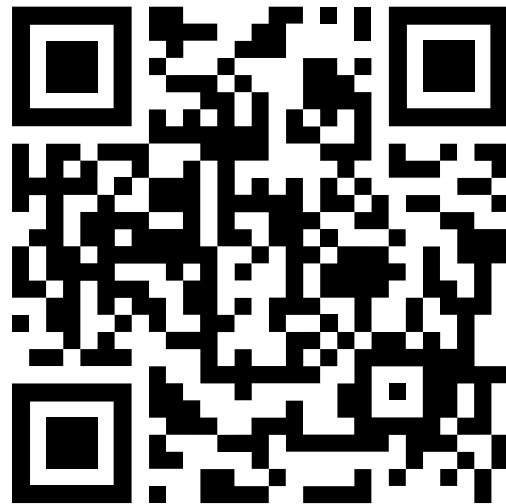


Contacts:

kirill.alekseev@corp.mail.ru

Slides

<https://vk.cc/cjCU20>



Co-organizer

Yandex